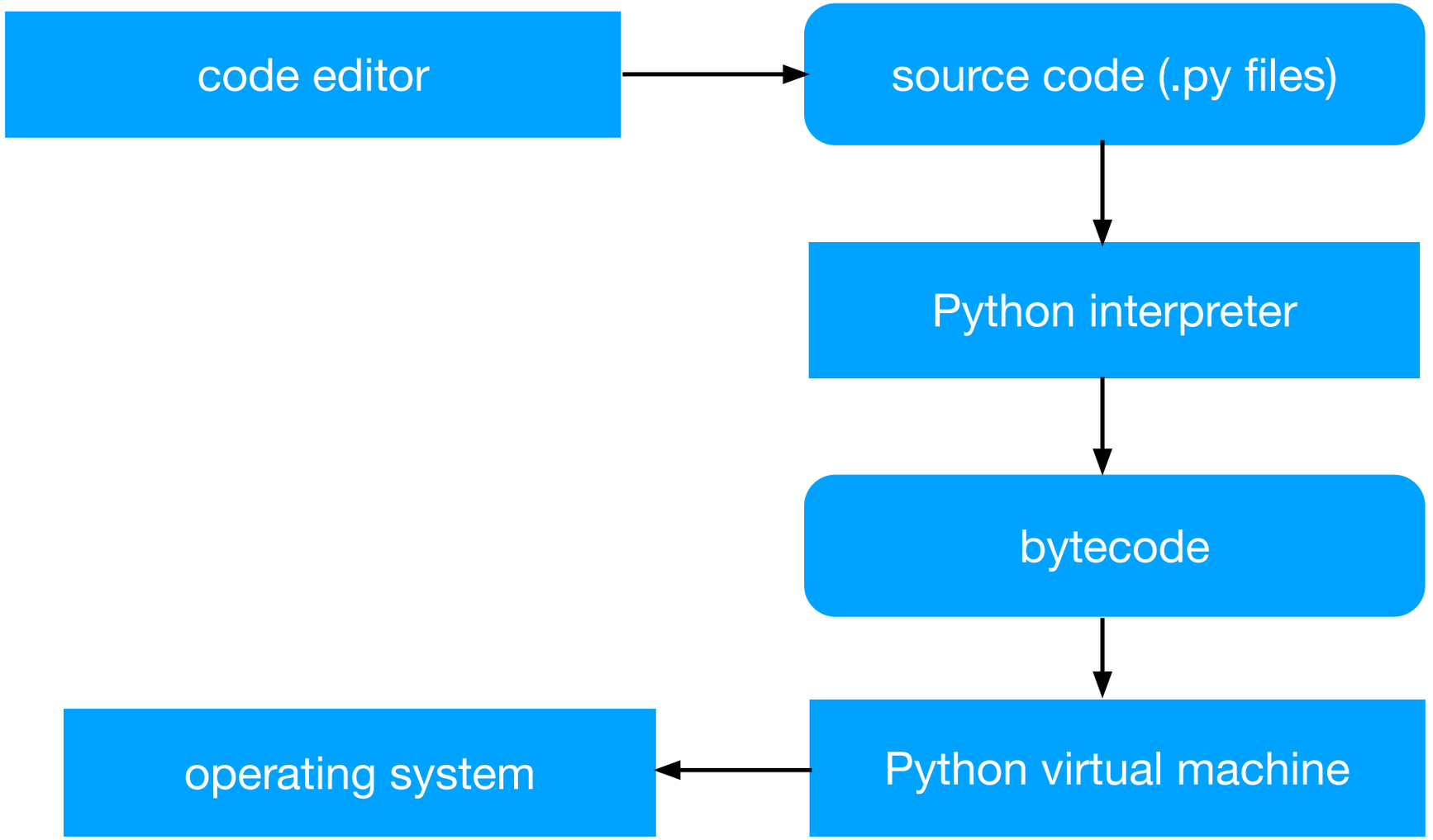
Python Fundamentals

Python Programming Language

- Python is a general-purpose programming language.
- Python can be used to develop console applications, games, web applications, and even for system administration.
- General Steps of developing and running a Python program
  - Use a text editor or IDE to enter source code and save it to a file with a .py extension.
  - Use the Python interpreter to compile the source code into bytecode
  - Use the Python virtual machine to translate the bytecode into machine instructions



Python Coding Rules

- Python relies on proper indentation.
- The standard indentation is four spaces.
- With implicit continuation, you can divide statements after parentheses, brackets, and braces, and before or after operators like plus or minus signs.
- with explicit continuation, you can use the \ character to divide statements anywhere on a line.

Integrated Development Environment (IDE) for Python

- The IDE is included with the Python distribution. It is called IDLE.
- You can use one of these features to run your Python codes:
  - Interactive Shell
    - To start IDLE to open an interactive shell
    - To close the shell, select File->Close
    - To reopen the shell, select Run->Python Shell
    - To use the shell, enter your Python code after the >>> prompt
  - Source files
    - To start the IDLE's editor, File->New, or File->Open
    - To use the editor, enter Python Code as you would do with other code editor.
    - To save your source file, select FIle->Save, or File->SaveAs
    - To compile and run the source file, select Run->Run Module or press the F5 key.

Data Types and Variables

- Basic Python data types
  - str e.g. “Peter”
  - int e.g 10000
  - float e.g 3.5
- To code a string constant, you can enclose it in a single or double quotation marks. e.g. “Peter” or ‘Peter’
- A variable can change as code executes.
- You can assign a value of any data type to a variable that has previously been assigned with a value of a different data type.
- Python is a case-sensitive language, so variable names are case-sensitive.
- Set all variables with a lowercase letter.
- Use underscore notation or camel case.
  - my_first_course, or myFirstCourse
- Don’t use the names of built-in function or Python keywords.
- When referring to non-object-oriented features, we suggest utilizing underscore notation (functions and global variables). Because camel case is widely used with object-oriented programming, we will use it for all of the object-oriented modules. But, your own applications can be written using any convention you like. The most important thing is to pick one of these conventions and follow it consistently for the duration of a module.

Python Division Operators

- Division operator /
  - 5 / 2 # 2.5
- Integer Division operator //
  - e.g. 5 // 2 # 2
- Module operator
  - e.g. 5 % 2 # 1

Strings

- To convert a number to a string, use str(number)
  - e.g str(price)
- To concatenate strings into one string, use the + operator
  - message = “My “ + course_name + “ is scheduled on “ + course_time

The print() function

- The print() function that prints a message onto the screen.
- Syntax: print(data[, sep=‘ ‘] [, end=‘\n’])
  - print(3.8) # 3.8
  - print(“GPA:”, 3.8) # GPA: 3.8
  - print(2, 4, 6, 8) # 2 4 6 8
- The following statements give the same result:
  - print(“GPA:”, gpa, “\nUnits:”, units)
  - print(“GPA: ” + gpa + “\nUnits: ” + units)
- Use optional the sep and end arguments
  - print(2,4,6,8,sep=‘ : ‘) # 2:4:6:8
  - print(2,4,6,8,end=‘ ... ‘) # 2 4 6 8...

The input() function

- Use the input() function to get user input from the keyboard.
- Syntax: input([prompt])
- The input() function always returns string data, even if you enter a number.
- The following examples give the same result

Example1:

```
first_name = input(“Enter your first name: “)
print(“Hello, “ + first_name + “!”)
```

Example2:

```
input(“Enter your first name: “)
first_name = input()
print(“Hello, “ + first_name + “!”)
```

The int(), float() and round() function

- The int() function converts the argument to the int type and returns the int value.
  - int(data)
- The float() function converts the argument to the float type and returns the float value.
- The round() function rounds the number to the number of decimal places as specified by the digits argument.
  - The round(number [,digits])

Example 1:

```
num_products = input("Enter the number of products: ")
num_products = int(num_products)

unit_price = input("Enter the unit price: ")
unit_price = float(unit_price)

unit_price = round(unit_price, 2)          # 123.45
```

Example - Converts gallons to liters

```
#!/usr/bin/env python3

# display a greeting message
print("The Gallons to Liters Program")
print()

# get input from the user
gallons = float(input("Enter number of gallons:\t"))

# convert to liters
liters = gallons * 3.7854
liters = round(liters, 2)

# format and display the result
print()
print("Liters:\t\t\t\t" + str(liters))
print()
print("Bye")
```

```
The Gallons to Liters Program
Enter number of gallons:        10
Liters:                         37.85
Bye
```

Exercises

- Problem specification: Find an average.

  Goal: Find the average of three numbers.

  Output: The user will enter three numbers interactively, n1, n2, and n3.

  Formula:

$$\text{Average} = \frac{n1 + n2 + n3}{3}$$

- Program sketch:
  1. Define real numbers n1, n2, n3 and Average.
  2. Print program titles.
  3. prompt the user to enter three numbers.
  4. Read n1, n2, and n3 and print them out again so that the user can verify that they were entered correctly.
  5. Add the three numbers and divide the sum by 3. Store the result in Average.
  6. Print the result with two decimal places.

- According to the specification, try to write a Python Program by yourself.

Exercises

- Problem specification: Find the square feet (Area) of a house.

  Goal: Find the square feet (Area) of a house.
  Output: The user will enter length and width interactively.
  Formula: area = length * area.

- Program sketch:
  1. Define integer numbers length, width and area.
  2. Print program titles.
  3. prompt the user to enter length and width.
  4. Read length, width and print them out again so that the user can verify that they were entered correctly.
  5. Multiply the two numbers. Store the result in area.
  6. Print the area.

- According to the specification, try to write a Python Program by yourself.

Logical Operators

- and - returns a True value if both expressions are True.
- or - Returns a True value of either expression is True.
- not - Reverses the value of a Boolean expression.
- The “and” and “or” operators only evaluate the second expression if necessary. They are called short-circuit operators.

Examples:
gpa >= 3.8 and units > 16

not age >= 21

age >= 21 and city == “Fremont” and state == “California”

String Comparisons

- The evaluation moves from left to right with numbers, coming before letters, and uppercase letters coming before lowercase letters.
  - “Peter” < “petter” # True
  - “Comp“ < “Computer” # True
  - “2” < “6” # True
  - “10” < “6” # True
- Use the lower() to convert uppercase letters to lowercase letters without changing the string itself.
- Use the upper() to convert lowercase letters to uppercase letters without changing the string itself.
  - "peter".upper() == "Peter".upper() # True

if statements

- To code an if clause, you code the if keyword followed by a Boolean expression, followed by a colon. Then, you code one or more indented statements starting on the next line.That block ends when the indentation ends.

```
if boolean expression:
    statements…
[elif boolean expression:
    statements…]
[else:
    statements…]
```

```
#!/usr/bin/env python3

# display the program title
print("The Letter Grade Calculator")
print()

# get user input
score = int(input("Enter score (0-100): "));

if score >= 90:
    grade = 'A'
elif score >= 80:
    grade = 'B'
elif score >= 70:
    grade = 'C'
elif score >= 60:
    grade = 'D'
else:
    grade = 'F'

# display the result
print("The grade is: " + grade)
print()
print("Bye")
```

Case Study - A warning notice based on average

- The notices program determines (1) a student's average based on three test scores and (2) the student's passing/failing status

```
# display the program title
print("The Notice program")
print()

# get user inputs
student_id = int(input("Enter a Student ID number: "))
test1 = int(input("Enter first score (0-100): "))
test2 = int(input("Enter second score (0-100): "))
test3 = int(input("Enter third score (0-100): "))
print()
print("Student number: ", student_id)
print("Test Scores: ", test1, test2, test3)

# calculate the average of three test grades and
# determine whether or not the student is passing
if test1 < 0 or test2 < 0 or test3 < 0:
        dataok = 0
else:
        dataok = 1

if dataok:
        average = (test1 + test2 + test3) / 3
        print("Average Score is : ", round(average,2))

        if average >= 60.0:
                notice = "Passing"
                if average < 70.0:
                        notice += " but marginal"
                print(notice + ".")
        else:
                print("Failing.")
else:
        print("Invalid Data: Scores(s) less than zero.\n")
print()
print("Bye")
```

The while loop statement

- The while loop uses a boolean condition to determine the number of times. If the condition is True, it executes the statements in the while loop until the condition is False.

```
#!/usr/bin/env python3
# display the program title
print("The Monthly Payment and Balance program")
print()

# get user inputs
amount = float(input("Enter amount of loan (e.g.
10000.00): "))
rate = float(input("Enter annual interest rate %: "))
payment = float(input("Enter monthly payment (e.g.
350.00): "))

# convert r to a monthly rate
rate /= (100 * 12)
balance = amount

m = 1
while balance > 0.0:
    print(m, '\t', round(balance,2))
    balance += rate*balance;    # add interest to remaining
balance
    balance -= payment;        # subtract monthly payment
    m += 1

print()
print('Bye!')
```

```
Enter amount of loan (e.g. 10000.00): 10000
Enter annual interest rate %: 5
Enter monthly payment (e.g. 350.00): 1000
1       10000.0
2       9041.67
3       8079.34
4       7113.0
5       6142.64
6       5168.24
7       4189.77
8       3207.23
9       2220.59
10       1229.84
11       234.97
Bye!
```

Example of finding average score

```
# display the program title
print("The Test Scores program")
print()

print("Enter 911 to end input")

# initialize variables
num_scores = 0
total_score = 0
test_score = 0

while True:
    test_score = int(input("Enter test score: "))
    if test_score >= 0 and test_score <= 100:
        total_score += test_score
        num_scores += 1
    elif test_score == 911:
        break
    else:
        print("Test score must be from 0 through 100."
            + "Score discarded. Try again.")

# calculate average score
if num_scores > 0:
        average_score = round(total_score / num_scores, 2)
else:
        average_score = 0.0

# format and display the result
print("======================")
print("Total Score:", total_score, "\nAverage Score:", average_score)
print()
print("Bye")
```

```
The Test Scores program
Enter 911 to end input
Enter test score: 100
Enter test score: 90
Enter test score: 90
Enter test score: 911
======================
Total Score: 280
Average Score: 93.33
Bye!
```

The for statements with range()

- You can use for statements to define for loops that run once for each integer in a a collection of integers returned by the range() functions.
- Syntax:

  for int var in range():
      statements...
- range(stop) - returns integer values from 0 to the stop value, but excluding the stop value.
- range(start, stop[, step]) - returns integer values from the start value to the stop value, but excluding the stop value. If the step value is specified, it increments or decrement by the step value. Examples,
  - range(5) # 0, 1, 2, 3, 4
  - range(2,4) # 2, 3
  - range(2,10,2) # 2, 4, 6, 8
  - range(10,0,-2) # 10, 8, 6, 4, 2
- The for loop examples:

  sum = 0
  for i in range(1,6)
      sum += i
  print("The sum of numbers from 1 to 5 =", sum)

```
#!/usr/bin/env python3
# display the program title

print("Print all even numbers from 0 to 10")
print()

for i in range(0,11,2):
      print(i)

print()
print('Bye!')
```

Example - Finding Prime Numbers

- The following program prints all the prime numbers from 2 to 20 using a nested “for” loop.

```
# display the program title

print("Find the prime numbers from 2 to 20")
print()

for i in range(2,20):
    isprime = True
    for j in range(2, i // 2 + 1):
        if i % j == 0:
            isprime = False
            break
    if isprime == True:
        print(i, "is prime")

print()
print('Bye!')
```

```
Find the prime numbers from 2 to 20
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
Bye!
```

Example - A Simple Savings Calculator

```
print("The Simple Savings Calculator")
print()

confirm = "y"
while confirm.lower() == "y":

    # get inputs from the keyboard
    initial_deposit_amount = float(input("Enter initial deposit amount:\t"))
    annual_interest_rate = float(input("Enter annual interest rate:\t"))
    years = int(input("Enter number of years:\t\t"))

    # convert annual interest rate to monthly interest rates
    monthly_interest_rate = annual_interest_rate / 12 / 100
    months = years * 12

    # calculate the future value
    estimated_balance = initial_deposit_amount
    for i in range(months):
        monthly_interest_amount = estimated_balance * monthly_interest_rate
        estimated_balance = estimated_balance + monthly_interest_amount

    # format and display the result
    print("Estimated Balance:\t\t" + "{:.2f}".format(estimated_balance))
    print()

    # see if the user wants to continue
    confirm = input("Continue? (y/n): ")
    print()

print("Bye!")
```

```
The Simple Savings Calculator
Enter initial deposit amount:   10000
Enter annual interest rate:     5
Enter number of years:          10
Estimated Balance:              16470.09
Continue? (y/n): y

Enter initial deposit amount:   20000
Enter annual interest rate:     2
Enter number of years:          10
Estimated Balance:              24423.99
Continue? (y/n): n

Bye!
```

Exercise - Convert C Program to Python

```
#include <stdio.h>

int main(void) {
   float sum;                 /* Sum of 12 rainfall amounts */
   char response;             /* User response ('y' or 'n') */
   float amount;              /* Rainfall amount for each month */
   int count;                 /* Loop control variable */

   do  {
      /* Input 12 monthly rainfall amounts. Verifying that each is nonnegative, and calculates the sum.*/
      sum = 0;
      for (count = 1; count <= 12; count++) {
         printf("Enter rainfall amount %d:", count);

         /* Inputs one month's rainfall amount. */
         do {
            scanf("%f", &amount);
            if (amount < 0.0f)
               printf("Amount cannot be negative. Enter again: ");
         } while (amount < 0.0f);
         sum = sum + amount;
      }
      printf("\n");
      printf("Average rainfall is %7.2f inches\n", sum / 12);
      printf("Do you have another recording site? (y or n): ");

      /* Inputs a character from the user and, if necessary, repeatedly prints an error message and inputs another
        character if the character isn't 'y' or 'n'.
      */
      do
      {
         scanf(" %c", &response);
         if (response != 'y' && response != 'n')
            printf("Please try y or n: ");

      } while (response != 'y' && response != 'n');
   } while (response == 'y');
   return 0;
}
```

Defining a Function

- A function is a block of code that can be called by another function or other statements.
- To define a function, code the the def keyword, the function name, a set of parameters in parentheses, and a colon. The below is the syntax:

  def function_name([parameters]):
  statements
- It is a good practice when coding functions in a program, you should also define a main() to start the program.
- When working with multiple modules, you may import one program as a module into another program. You can add a if statement to check if the current module is being run as the main module. It yes, it calls the main function.

Example - A Simple Savings Calculator in Functions

```
def calculate_savings(initial_deposit_amount, annual_interest_rate, years):
    # convert annual interest rate to monthly interest rates
    monthly_interest_rate = annual_interest_rate / 12 / 100
    months = years * 12

    # calculate the future value
    estimated_balance = initial_deposit_amount
    for i in range(months):
        monthly_interest_amount = estimated_balance * monthly_interest_rate
        estimated_balance = estimated_balance + monthly_interest_amount
    return estimated_balance

def main():
    print("The Simple Savings Calculator using Functions")
    print()

    confirm = "y"
    while confirm.lower() == "y":
        # get inputs from the keyboard
        initial_deposit_amount = float(input("Enter initial deposit amount:\t"))
        annual_interest_rate = float(input("Enter annual interest rate:\t"))
        years = int(input("Enter number of years:\t\t"))

        # get the estimated balance
        estimated_balance = calculate_savings(initial_deposit_amount, annual_interest_rate, years)

        # format and display the result
        print("Estimated Balance:\t\t" + "{:.2f}".format(estimated_balance))
        print()
        # see if the user wants to continue
        confirm = input("Continue? (y/n): ")
        print()

    print("Bye!")

if __name__ == "__main__":
    main()
```

Default Arguments and Named Arguments

- You can specify a default value for any parameters in a function. However, the default parameters with default values must be coded last in the parameter list.
  def calculate_savings(initial_deposit_amount, annual_interest_rate, years = 10): # correct
  def calculate_savings(initial_deposit_amount, annual_interest_rate=2.0, years = 10): # correct
  def calculate_savings(initial_deposit_amount, annual_interest_rate=2.0, years): # incorrect
- To use default argument examples:
  estimated_balance = calculate_savings(10000, 5)
  same as
  estimated_balance = calculate_savings(10000, 5, 10)
- If you can call a function with named arguments, you don't have to code the arguments in the same sequence of the function parameters.
  estimated_balance = calculate_savings(annual_interest_rate=5.0, years=30, initial_deposit_amount=20000)
- Using named arguments in calling a function can improve the readability of the code.

Example - A Simple Savings Calculator using default arguments and named arguments

```
#!/usr/bin/env python3

def calculate_savings(initial_deposit_amount, annual_interest_rate=2.0, years = 10):
        # convert annual interest rate to monthly interest rates
        monthly_interest_rate = annual_interest_rate / 12 / 100
        months = years * 12

        # calculate the future value
        estimated_balance = initial_deposit_amount
        for i in range(months):
                monthly_interest_amount = estimated_balance * monthly_interest_rate
                estimated_balance = estimated_balance + monthly_interest_amount
        return estimated_balance

def main():
        print("The Simple Savings Calculator using default arguments and named arguments")
        print()

        # using default arguments
        estimated_balance = calculate_savings(10000, 2.0, 10)
        print("calculate_savings(10000, 2.0, 10):\t" + "{:.2f}".format(estimated_balance))

        estimated_balance = calculate_savings(10000, 2.0)
        print("calculate_savings(10000, 2.0):\t\t" + "{:.2f}".format(estimated_balance))

        estimated_balance = calculate_savings(10000)
        print("calculate_savings(10000):\t\t" + "{:.2f}".format(estimated_balance))

        estimated_balance = calculate_savings(annual_interest_rate=5.0, years=30, initial_deposit_amount=20000)
        print("calculate_savings(annual_interest_rate=5.0, years=30, initial_deposit_amount=20000):\t" +
"{:.2f}".format(estimated_balance))

        print()
        print("Bye!")

if __name__ == "__main__":
        main()
```

Local and Global Variables

- A local variable is defined inside a function.

```
def cal_sum(x, y):
    sum = x + y                 # sum is a local variable
    return sum
```

- A global variable is defined outside of all functions.

```
sum = 0
def cal_sum(x, y):
    sum = x + y                 # sum is a global variable
    return sum
```

- When a local variable has the same name as a global variable. the local variable overrides the global variable.
- By default, a global variable is read only inside a function.
- In order to modify a global variable in a function, you need to use the global keyword to declare it.
- Since the global variable is read only be default in a function, we can use a global variable as a global constant. Normally, we define global constants in a different module.

```
PI = 3.14               # Global Constant
```

Example of Global Variables

```
PI = 3.14                    # Global Constant
sum = 0                      # Gloabl Variable

def calsum(x, y):
        sum = x + y
        return sum

def calsum2(x, y):
        global sum
        sum = x + y
        return sum

def printsum():
        global sum
        print('global sum =', sum, '\n')

def getarea(radius):
        return radius * radius * PI

def main():
        print("The Demo Program for Global Variables\n")

        total = calsum(10, 20)
        print('calsum(10, 20) =', total)
        printsum()

        total = calsum2(10, 20)
        print('calsum2(10, 20) =', total)
        printsum()

        area = getarea(8)
        print('getarea(10) =', round(area,2), '\n')

        print("Bye!")

if __name__ == "__main__":
        main()
```

The Demo Program for Global Variables

calsum(10, 20) = 30
global sum = 0

calsum2(10, 20) = 30
global sum = 30

getarea(10) = 200.96

Bye!

Python Module

- A module is file that contains reusable code such as functions
- The name of the module is the name of the file.
- The syntax for importing a module
  import moduleName [as namespace]
  - Example of importing into the default namespace
    import unitconverters
    g = unitconverters.to_gallon(1)
  - Example of importing into a specified namespace
    import unitconverters as converters
    g = converters.to_gallon(1)
- The syntax for importing functions into the global namespace.
  from moduleName import function1[, function2]...
  - Example 1
    from unitconverters import to_gallon
    g = to_gallon(1)
  - Example 2
    *from unitconverters import **
    g = to_gallon(1)
    l = to_liter(2)

unitconverters.py

```
#This module contains functions for converting
liquid capacity between gallons and degrees
liters

def to_liter(gallons):
        # Accepts gallons and returns liters
        liters = gallons * 3.785411784
        return liters

def to_gallon(liters):
        # Accepts liters and returns gallons
        gallons = liters * 0.2641720524
        return gallons

def main():
        print("1 gallon =", to_liter(1), "liters")
        print("1 liter =", to_gallon(1), "gallons")

if __name__ == "__main__":
        main()
```

Example of using module

```
import unitconverters as converter

def display_menu():
        print("The Convert Liquid Capacity Program")
        print()
        print("MENU")
        print("1. Gallon to Liter")
        print("2. Liter to Gallon")
        print()

def convert_liquid_cap():
        option = int(input("Enter a menu option: "))
        if option == 1:
                g = float(input("Enter gallons: "))
                l = converter.to_liter(g)
                l = round(l, 2)
                print("Liters:", l)
        elif option == 2:
                l = float(input("Enter liters: "))
                g = converter.to_gallon(l)
                g = round(g, 2)
                print("Gallons:", g)
        else:
                print("You enterred an invalid menu number.")

def main():
        display_menu()
        again = "y"
        while again.lower() == "y":
                convert_liquid_cap()
                print()
                again = input("Convert another liquid capacity unit?
(y/n): ")
                print()
        print("Bye!")

if __name__ == "__main__":
        main()
```

```
The Convert Liquid Capacity Program
MENU
1. Gallon to Liter
2. Liter to Gallon
Enter a menu option: 1
Enter gallons: 1
Liters: 3.79
Convert another liquid capacity unit? (y/n): n
Bye!
```

Module Documentation

- A docstring begins and ends with three double quotes

unitconverters.py

```
"""
This module contains functions for converting liquid capacity
between gallons and degrees liters
"""

def to_liter(gallons):
    """
    Accepts gallons and returns liters
    """
    liters = gallons * 3.785411784
    return liters


def to_gallon(liters):
    """
    Accepts liters and returns gallons
    """
    gallons = liters * 0.2641720524
    return gallons

def main():
    print("1 gallon =", to_liter(1), "liters")
    print("1 liter =", to_gallon(1), "gallons")

if __name__ == "__main__":
    main()
```

Using Python interactive shell to view the documentation

```
>>> import unitconverters
>>> help(unitconverters)
Help on module unitconverters:
NAME
   unitconverters
DESCRIPTION
   This module contains functions for converting liquid capacity
   between gallons and degrees liters
FUNCTIONS
   main()

   to_gallon(liters)
      Accepts liters and returns gallons

   to_liter(gallons)
      Accepts gallons and returns liters
FILE
   /home/kcheung/apps/aws/cs487/week02/unitconverters.py
```

Example of using standard module

```
#!/usr/bin/env python3

import random

LIMIT = 100

# get a random numbe form 1 to 100
magic = random.randint(1, LIMIT)
count = 0

while True:
    guess = int(input("Enter your guess (1 to 100): "))
    if guess < magic:
        print("Too low.")
        count += 1
    elif guess > magic:
        print("Too high.")
        count += 1
    elif guess == magic:
        print("You guessed it in " + str(count) + " tries.\n")
        break;
print('bye!')
```

```
Enter your guess (1 to 100): 50
Too high.
Enter your guess (1 to 100): 25
Too high.
Enter your guess (1 to 100): 12
Too high.
Enter your guess (1 to 100): 8
Too low.
Enter your guess (1 to 100): 9
Too low.
Enter your guess (1 to 100): 10
You guessed it in 5 tries.
bye!
```